Functional Synthesis: An Ideal Meeting Ground for Formal Methods and Machine Learning

Kuldeep S. Meel [1]

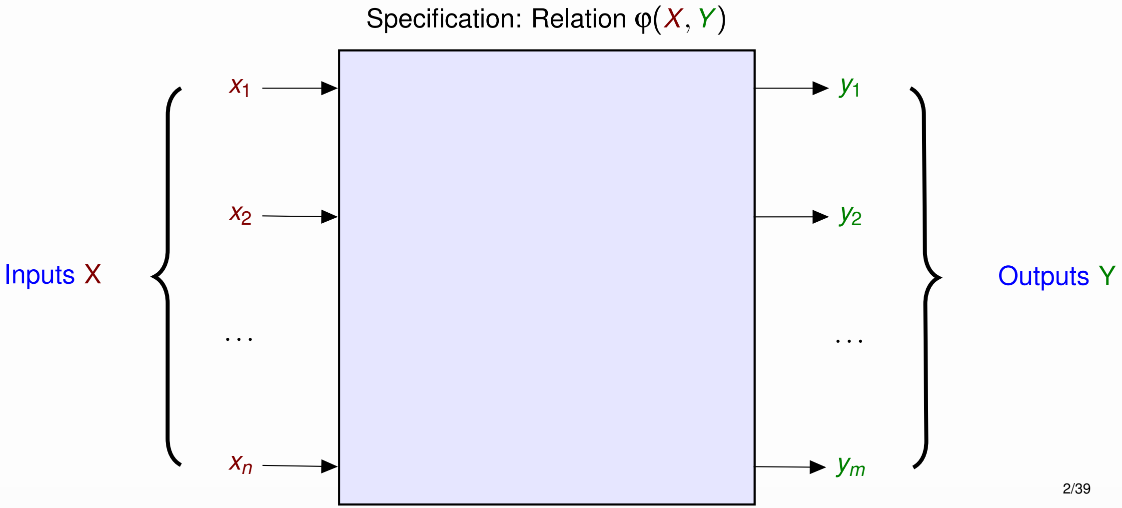
Joint work with: Priyanka Golia [1,2] and Subhajit Roy [2]

[1] National University of Singapore
[2] Indian Institute of Technology Kanpur

Corresponding Papers: CAV-20, IJCAI-21, ICCAD-21

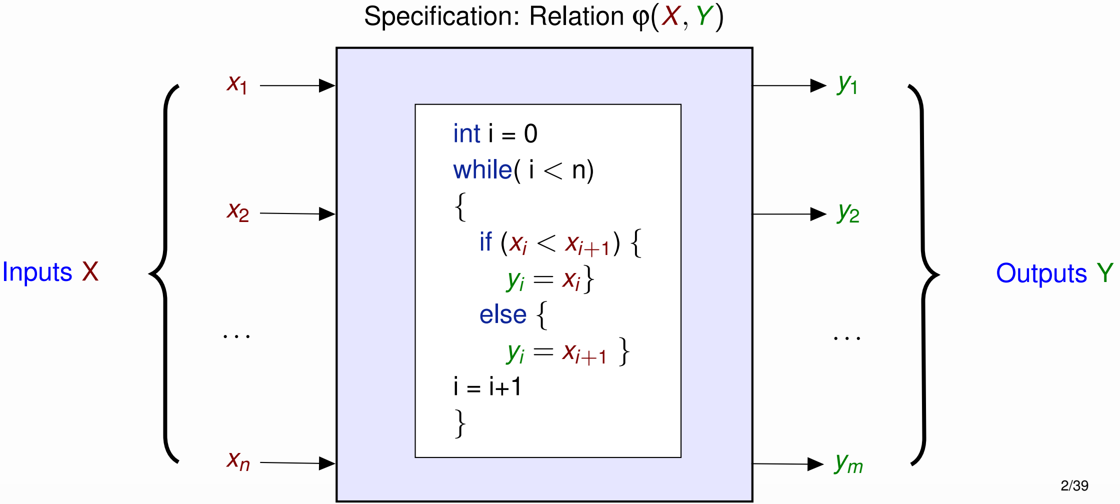
Synthesis

Holy Grail of Programming: *The user states the problem, the computer solves it* (Freuder, 1996)



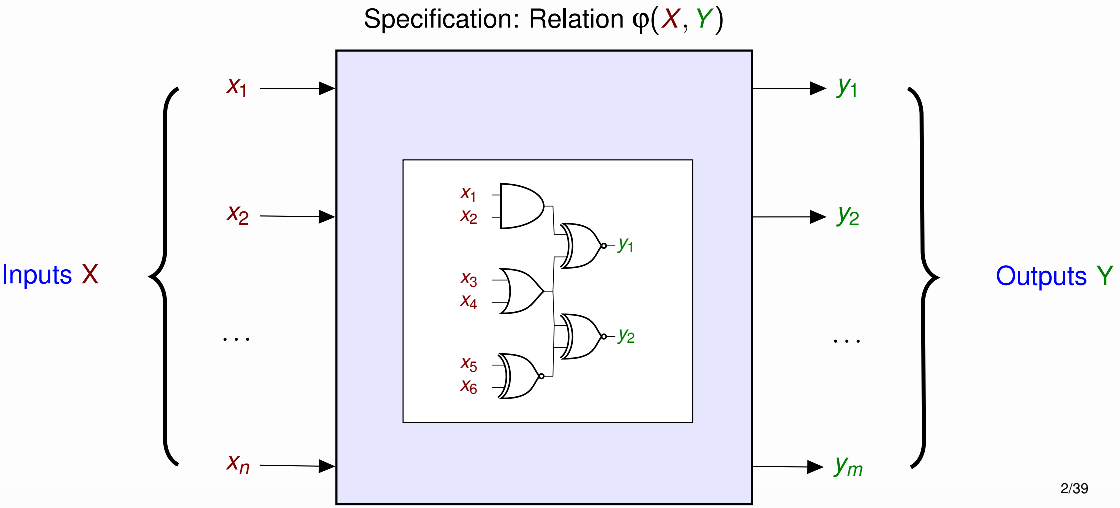
Synthesis

Holy Grail of Programming: *The user states the problem, the computer solves it* (Freuder, 1996)



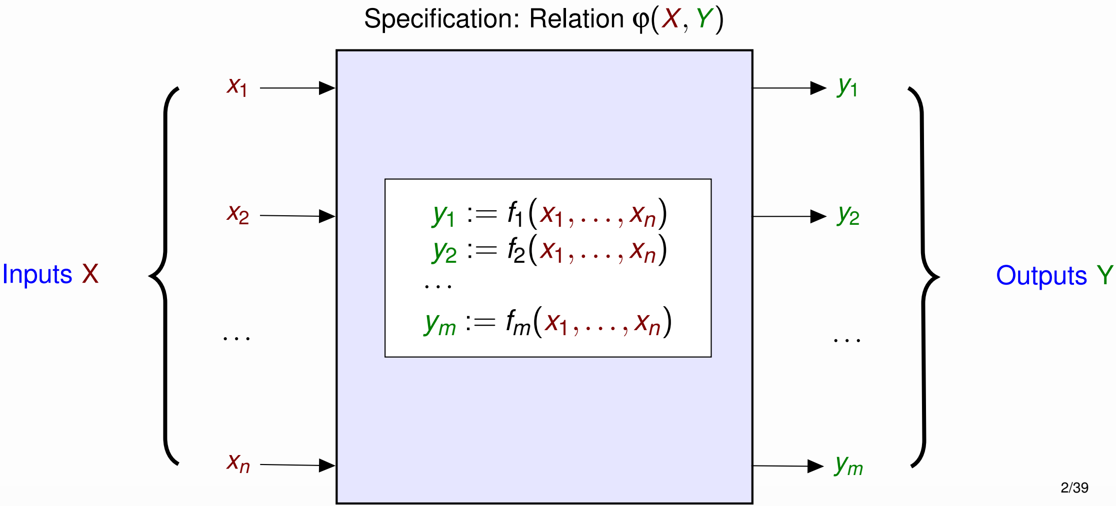
Synthesis

Holy Grail of Programming: *The user states the problem, the computer solves it* (Freuder, 1996)



Synthesis

Holy Grail of Programming: *The user states the problem, the computer solves it* (Freuder, 1996)



Source of Specifications: Program Synthesis

$$
\begin{aligned}
&g_1(x_1, x_2) \geq x_1 \text{ and} \\
&g_1(x_1, x_2) \geq x_2 \text{ and} \\
&(g_1(x_1, x_2) == x_1 \text{ or} \\
&g_1(x_1, x_2) == x_2)
\end{aligned}
$$

Sythesise a function g_1
that satisfies the specification

Golia, Roy, and M. (IJCAI-21)

Source of Specifications: Program Synthesis

$g_1(x_1, x_2) \geq x_1$ and
$g_1(x_1, x_2) \geq x_2$ and
$(g_1(x_1, x_2) == x_1$ or
$g_1(x_1, x_2) == x_2)$

Introduce variable y_1
Replace $g_1(x_1, x_2)$ call by y_1

$y_1 \geq x_1$ and
$y_1 \geq x_2$ and
$(y_1 == x_1$ or
$y_1 == x_2)$

Sythesise a function g_1
that satisfies the specification

Golia, Roy, and M. (IJCAI-21)

Functional Synthesis

Given $\varphi(X, Y)$ over inputs $X = \{x_1, x_2, \ldots, x_n\}$ and outputs $Y = \{y_1, y_2, \ldots, y_m\}$.

Synthesize A function vector $F = \{f_1, f_2, \ldots, f_m\}$, such that $y_i := f_i(x_1, \ldots, x_n)$ such that:

$$\exists Y \varphi(X, Y) \equiv \varphi(X, F(X))$$

Each f_i is called Skolem function and F is called Skolem function vector.

Key Challenge: $\varphi(X, Y)$ is a relation

Non-uniqueness of Skolem Functions

Let $X = \{x_1, x_2\}$, $Y = \{y_1\}$ and $\varphi(X, Y) = x_1 \vee x_2 \vee y_1$

Possible Skolem function: $f(x_1, x_2) := \neg(x_1 \vee x_2)$

Non-uniqueness of Skolem Functions

Let $X = \{x_1, x_2\}$, $Y = \{y_1\}$ and $\varphi(X, Y) = x_1 \vee x_2 \vee y_1$

Possible Skolem function: $f(x_1, x_2) := \neg(x_1 \vee x_2)$

$$\varphi(X, F(X)) = x_1 \vee x_2 \vee (\neg(x_1 \vee x_2))$$

X		$\exists Y \varphi(X, Y)$	$\varphi(X, F(X))$
$x_1 = 0, x_2 = 0$	$y_1 = 1$	True	True
$x_1 = 0, x_2 = 1$	$y_1 = 1$	True	True
$x_1 = 1, x_2 = 0$	$y_1 = 1$	True	True
$x_1 = 1, x_2 = 1$	$y_1 = 1$	True	True

$$\exists Y \varphi(X, Y) \equiv \varphi(X, F(X))$$

Non-uniqueness of Skolem Functions

Let $X = \{x_1, x_2\}$, $Y = \{y_1\}$ and $\varphi(X, Y) = x_1 \vee x_2 \vee y_1$

Possible Skolem function: $f(x_1, x_2) := \neg(x_1 \vee x_2)$

$$\varphi(X, F(X)) = x_1 \vee x_2 \vee (\neg(x_1 \vee x_2))$$

X	$\exists Y \varphi(X, Y)$		$\varphi(X, F(X))$
$x_1 = 0, x_2 = 0$	$y_1 = 1$	True	True
$x_1 = 0, x_2 = 1$	$y_1 = 1$	True	True
$x_1 = 1, x_2 = 0$	$y_1 = 1$	True	True
$x_1 = 1, x_2 = 1$	$y_1 = 1$	True	True

$$\exists Y \varphi(X, Y) \equiv \varphi(X, F(X))$$

Other possible Skolem functions: $f_1(x_1, x_2) = \neg x_1$ $\quad f_1(x_1, x_2) = \neg x_2$ $\quad f_1(x_1, x_2) = 1$

The Many Forms of Functional Synthesis

Functional synthesis is also

- *Church's Problem* (Circuit Synthesis) for Propositional Logic
- Program synthesis for propositional logic
  - No restrictions on the grammar
- *Strategy Synthesis* $\varphi(X, Y)$
  - X player is trying to falsify φ while Y player is trying to satisfy φ

Diverse Approaches

- From the proof of validity of $\forall X \exists Y \varphi(X, Y)$
  - (Bendetti et al., 2005)
  - (Jussilla et al., 2007)
  - (Heule et al., 2014)
- Quantifier instantiation in SMT solvers
  - (Barrett et al., 2015)
  - (Bierre et al.,2017)
- Input-Output Separation
  - (Chakraborty et al., 2018)
- Knowledge representation
  - (Kukula et al., 2000)
  - (Trivedi et al., 2003)
  - (Jiang, 2009)
  - (Kuncak et al., 2010)
  - (Balabanov and Jiang, 2011)
  - (John et al., 2015)
  - (Fried, Tabajara, Vardi, 2016,2017)
  - (Akshay et al., 2017,2018)
  - (Chakraborty et al., 2019)
- Incremental determinization
  - (Rabe et al., 2015, 2018, 2019)

Diverse Approaches

- From the proof of validity of $\forall X \exists Y \varphi(X, Y)$

  (Bendetti et al., 2005)
  (Jussilla et al., 2007)
  (Heule et al., 2014)

- Quantifier instantiation in SMT solvers

  (Barrett et al., 2015)
  (Bierre et al.,2017)

- Input-Output Separation

  (Chakraborty et al., 2018)

- Knowledge representation

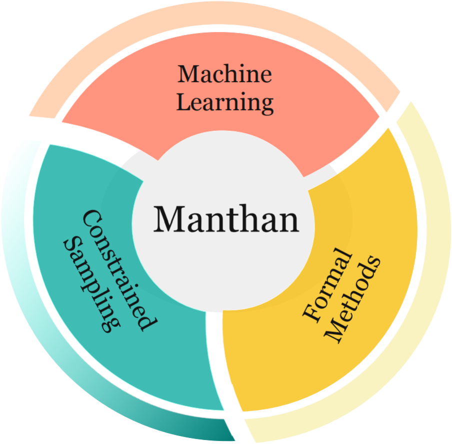
  (Kukula et al., 2000)
  (Trivedi et al., 2003)
  (Jiang, 2009)
  (Kuncak et al., 2010)
  (Balabanov and Jiang, 2011)
  (John et al., 2015)
  (Fried, Tabajara, Vardi, 2016,2017)
  (Akshay et al., 2017,2018)
  (Chakraborty et al., 2019)

- Incremental determinization

  (Rabe et al., 2015, 2018, 2019)

Scalability remains the holy grail

A Data-Driven Approach for Boolean Functional Synthesis

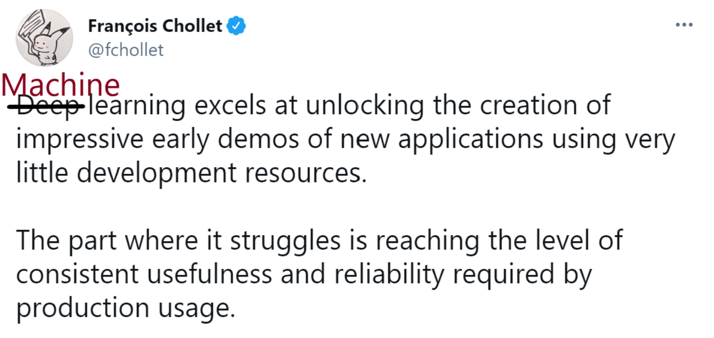


Take Away Message

François Chollet
@fchollet

Machine ~~Deep~~ learning excels at unlocking the creation of impressive early demos of new applications using very little development resources.

The part where it struggles is reaching the level of consistent usefulness and reliability required by production usage.

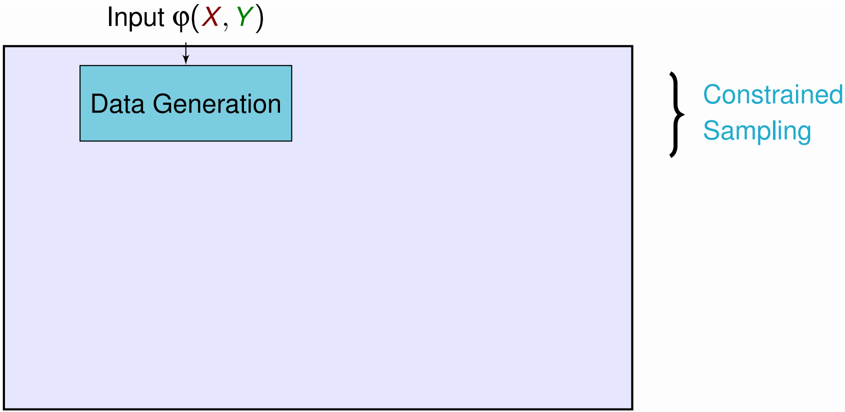


François Chollet
@fchollet

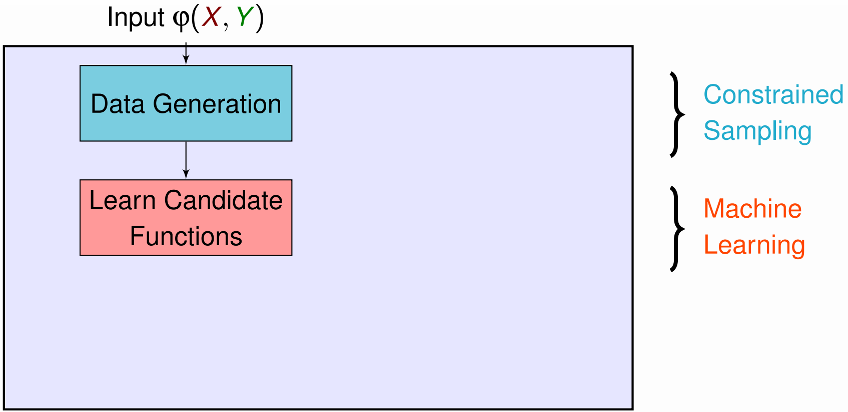
Machine ~~Deep~~ learning excels at unlocking the creation of impressive early demos of new applications using very little development resources.

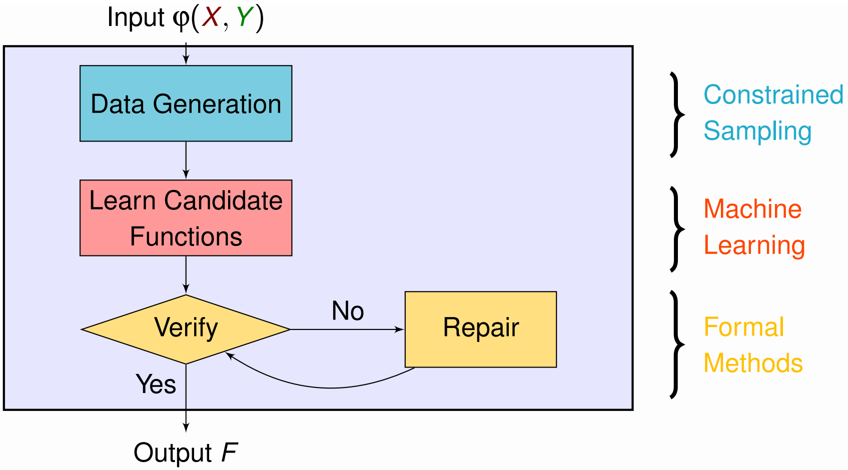
The part where it struggles is reaching the level of consistent usefulness and reliability required by production usage.

Formal Methods is the Answer to Machine Learning's Struggles


Input φ(X, Y)
Data Generation
Constrained Sampling

Manthan


Input φ(X, Y)
Data Generation
Learn Candidate Functions
Constrained Sampling
Machine Learning


Input φ(X, Y)
Data Generation
Learn Candidate Functions
Verify
No
Repair
Yes
Output F
Constrained Sampling
Machine Learning
Formal Methods

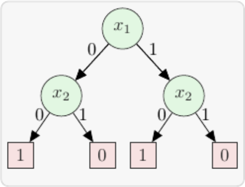
$\varphi(x_1, x_2, y_1, y_2)$ $\longrightarrow$

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0

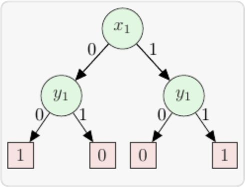
Learn Candidate Functions

Taming the Curse of Abstractions via Learning with Errors

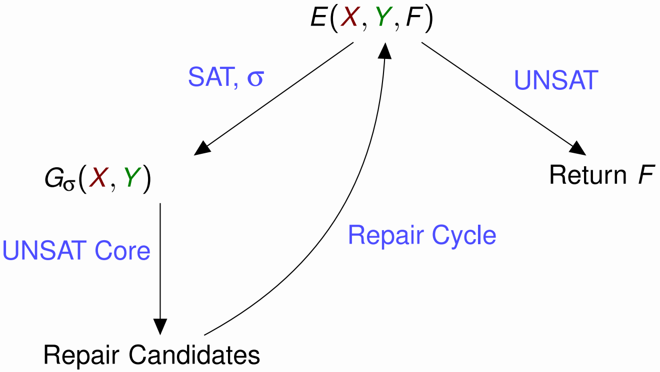
x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0



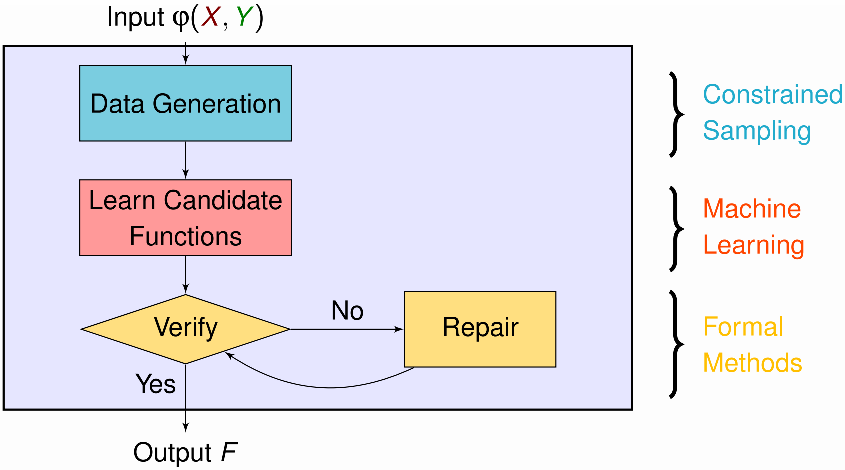
$p_1 := (\neg x_1 \wedge \neg x_2)$,
$p_2 := (x_1 \wedge \neg x_2)$
f_1 = if p_1 then 1
elif p_2 then 1
else 0



$p_1 := (\neg x_1 \wedge \neg y_1)$,
$p_2 := (x_1 \wedge y_1)$
f_1 = if p_1 then 1
elif p_2 then 1
else 0



$E(X, Y, F)$
SAT, σ
UNSAT
$G_\sigma(X, Y)$
Return F
UNSAT Core
Repair Cycle
Repair Candidates


Input φ(X, Y)
Data Generation
Learn Candidate Functions
Verify
No
Repair
Yes
Output F
Constrained Sampling
Machine Learning
Formal Methods

Data Generation

Potential Strategy: Randomly sample satisfying assignment of $\varphi(X, Y)$.

Challenge: Multiple valuations of y_1, y_2 for same valuation of x_1, x_2.

Data Generation

Potential Strategy: Randomly sample satisfying assignment of $\varphi(X, Y)$.

Challenge: Multiple valuations of y_1, y_2 for same valuation of x_1, x_2.

$$\varphi(x_1, x_2, y_1, y_2) : (x_1 \vee x_2 \vee y_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg y_2)$$

x_1	x_2	y_1	y_2
0	0	1	0/1
0	1	0/1	0/1
1	0	0/1	0/1
1	1	0/1	0

Data Generation

$$\varphi(x_1, x_2, y_1, y_2) : (x_1 \vee x_2 \vee y_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg y_2)$$

x_1	x_2	y_1	y_2
0	0	1	0/1
0	1	0/1	0/1
1	0	0/1	0/1
1	1	0/1	0

Uniform Sampler →

x_1	x_2	y_1	y_2
0	0	1	1
0	1	0	1
1	0	0	1
1	1	0	0

$$\varphi(x_1, x_2, y_1, y_2) : (x_1 \vee x_2 \vee y_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg y_2)$$

x_1	x_2	y_1	y_2
0	0	1	0/1
0	1	0/1	0/1
1	0	0/1	0/1
1	1	0/1	0

Uniform Sampler →

x_1	x_2	y_1	y_2
0	0	1	1
0	1	0	1
1	0	0	1
1	1	0	0

- Possible Skolem functions:
  - $f_1(x_1, x_2) = \neg(x_1 \vee x_2)$
  - $f_2(x_1, x_2) = \neg(x_1 \wedge x_2)$

Data Generation

$$\varphi(x_1, x_2, y_1, y_2) : (x_1 \lor x_2 \lor y_1) \land (\neg x_1 \lor \neg x_2 \lor \neg y_2)$$

x_1	x_2	y_1	y_2
0	0	1	0/1
0	1	0/1	0/1
1	0	0/1	0/1
1	1	0/1	0

Uniform Sampler →

x_1	x_2	y_1	y_2
0	0	1	1
0	1	0	1
1	0	0	1
1	1	0	0

- Possible Skolem functions:
  - $f_1(x_1, x_2) = \neg(x_1 \lor x_2)$ $\quad f_1(x_1, x_2) = \neg x_1$ $\quad f_1(x_1, x_2) = \neg x_2$ $\quad f_1(x_1, x_2) = 1$
  - $f_2(x_1, x_2) = \neg(x_1 \land x_2)$ $\quad f_2(x_1, x_2) = \neg x_1$ $\quad f_2(x_1, x_2) = \neg x_2$ $\quad f_2(x_1, x_2) = 0$

Data Generation

$$\varphi(x_1, x_2, y_1, y_2) : (x_1 \vee x_2 \vee y_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg y_2)$$

x_1	x_2	y_1	y_2
0	0	1	0/1
0	1	0/1	0/1
1	0	0/1	0/1
1	1	0/1	0

Magical Sampler →

x_1	x_2	y_1	y_2
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	0

- Possible Skolem functions:
  - $f_1(x_1, x_2) = \neg(x_1 \vee x_2)$ $\quad f_1(x_1, x_2) = \neg x_1$ $\quad f_1(x_1, x_2) = \neg x_2$ $\quad f_1(x_1, x_2) = 1$
  - $f_2(x_1, x_2) = \neg(x_1 \wedge x_2)$ $\quad f_2(x_1, x_2) = \neg x_1$ $\quad f_2(x_1, x_2) = \neg x_2$ $\quad f_2(x_1, x_2) = 0$

Weighted Sampling to Rescue

- $W : X \cup Y \mapsto [0, 1]$

- The probability of generation of an assignment is proportional to its weight.

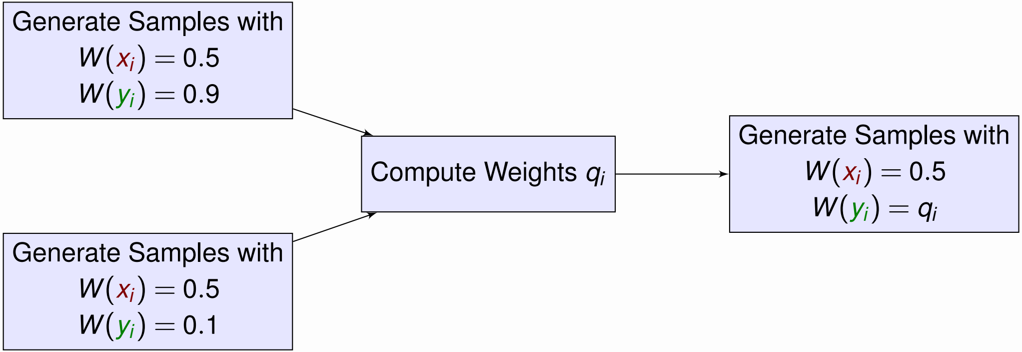
$$W(\sigma) = \prod_{\sigma(z_i)=1} W(z_i) \prod_{\sigma(z_i)=0} (1 - W(z_i))$$

- Example: $W(x_1) = 0.5 \quad W(x_2) = 0.5 \quad W(y_1) = 0.9 \quad W(y_2) = 0.1$
  $\sigma_1 = \{x_1 \mapsto 1, x_2 \mapsto 0, y_1 \mapsto 0, y_2 \mapsto 1\}$

$$W(\sigma_1) = 0.5 \times (1 - 0.5) \times (1 - 0.9) \times 0.1 = 0.0025$$

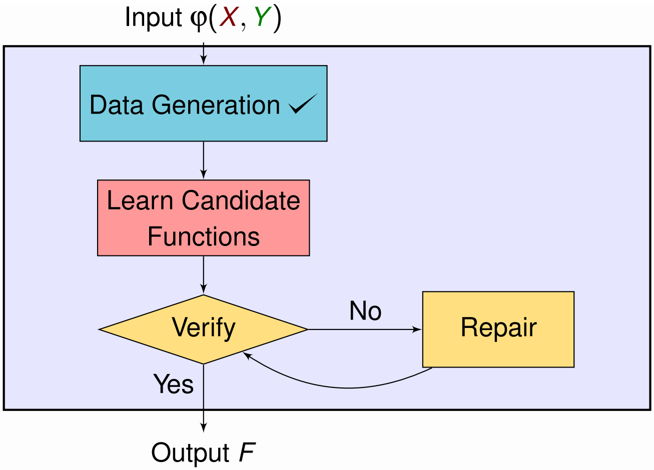
- Uniform sampling is a special case where all variables are assigned weight of 0.5.

Data Generation



Different Sampling Strategies

- Knowledge representation based techniques
  (Yuan,Shultz, Pixley,Miller,Aziz 1999)
  (Yuan,Aziz, Pixley,Albin, 2004)
  (Kukula and Shiple, 2000)
  (Sharma, Gupta, M., Roy, 2018)
  (Gupta, Sharma, M., Roy, 2019)
- Hashing based techniques
  (Chakraborty, M., and Vardi 2013, 2014,2015)
  (Soos, M., and Gocht 2020)
- Mutation based techniques
  (Dutra, Laeufer, Bachrach, Sen, 2018)
- Markov Chain Monte Carlo based techniques
  (Wei and Selman,2005)
  (Kitchen,2010)
- Constraint solver based techniques
  (Ermon, Gomes, Sabharwal, Selman,2012)
- Belief networks based techniques
  (Dechter, Kask, Bin, Emek,2002)
  (Gogate and Dechter,2006)


Input φ(X, Y)
Data Generation ✓
Learn Candidate Functions
Verify
No
Repair
Yes
Output F

Learn Candidate Function: Decision Tree Classifier

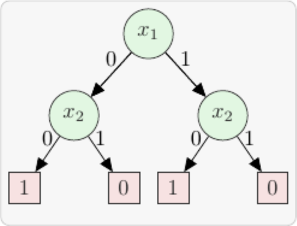
$$\varphi(x_1, x_2, y_1, y_2) : (x_1 \vee x_2 \vee y_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg y_2)$$

- To learn y_2
  - Feature set: valuation of x_1, x_2, y_1
  - Label: valuation of y_2
  - Learn decision tree to represent y_2 in terms of x_1, x_2, y_1
- To learn y_1
  - Feature set: valuation of x_1, x_2
  - Label: valuation of y_1
  - Learn decision tree to represent y_1 in terms of x_1, x_2

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0

Learning Candidate Functions

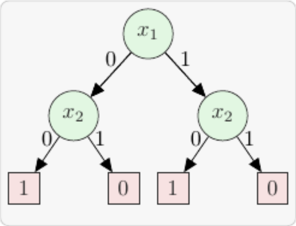
x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0



$p_1 := (\neg x_1 \wedge \neg x_2)$,
$p_2 := (x_1 \wedge \neg x_2)$
f_1 = if p_1 then 1
elif p_2 then 1
else 0

Learning Candidate Functions

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0



$p_1 := (\neg x_1 \wedge \neg x_2)$,

$p_2 := (x_1 \wedge \neg x_2)$

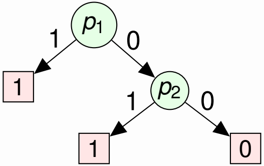
f_1 = if p_1 then 1

elif p_2 then 1

else 0

Can reorder p_1, p_2

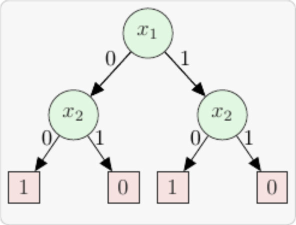
Learning one level decision list



What Kind of Learning

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0

→



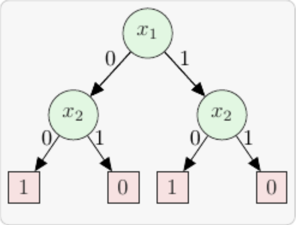
$p_1 := (\neg x_1 \wedge \neg x_2)$,
$p_2 := (x_1 \wedge \neg x_2)$
f_1 = if p_1 then 1
elif p_2 then 1
else 0

Learning without Error
Every row is a solution of $\varphi(X, Y)$

Learning with Errors
The data is only a subset of solutions.

What Kind of Learning

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0



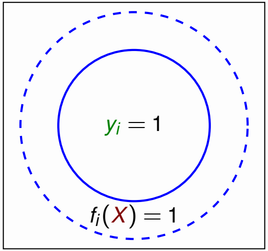
$p_1 := (\neg x_1 \wedge \neg x_2),$
$p_2 := (x_1 \wedge \neg x_2)$
$f_1 =$ if p_1 then 1
elif p_2 then 1
else 0

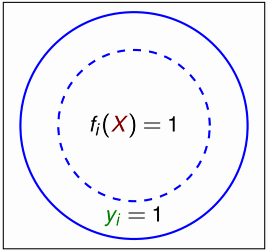
Learning without Error
Every row is a solution of $\varphi(X, Y)$

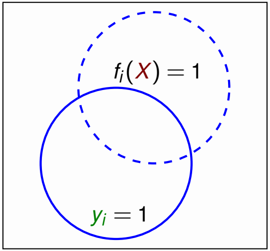
Learning with Errors
The data is only a subset of solutions.

Learn with Errors: Approximations <u>not</u> Abstractions

Abstraction vs Approximation







$y_i \to f_i(X)$

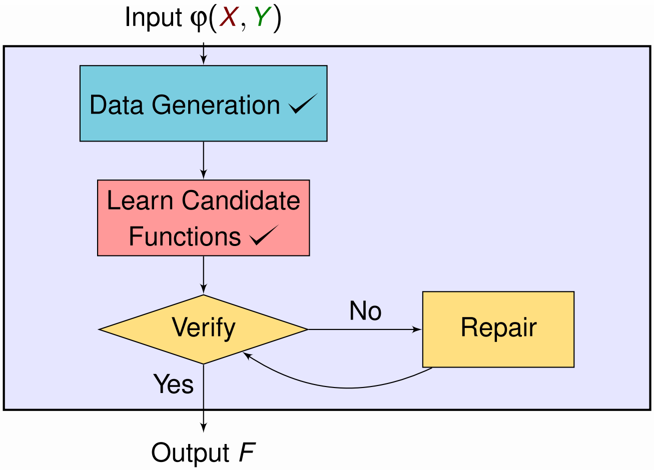
$f_i(X) \to y_i$

Abstraction

Approximation

$y_i = 1, f_i(X) = 0$

$y_i = 0, f_i(X) = 1$


Input φ(X, Y)
Data Generation ✓
Learn Candidate Functions ✓
Verify
No
Repair
Yes
Output F

Verification of Candidate Functions

$$E(X, Y, Y') := \varphi(X, Y) \wedge \neg\varphi(X, Y') \wedge (Y' \leftrightarrow F(X))$$

(JSCTA'15)

- If $E(X, Y, Y')$ is UNSAT: $\exists Y \varphi(X, Y) \equiv \varphi(X, F(X))$
  - Return F
- If $E(X, Y, Y')$ is SAT: $\exists Y \varphi(X, Y) \not\equiv \varphi(X, F(X))$
  - Let $\sigma \models E(X, Y, Y')$ be a counterexample to fix.

Repair Candidate Identification

$$E(X, Y, Y') := \varphi(X, Y) \wedge \neg\varphi(X, Y') \wedge (Y' \leftrightarrow F(X))$$

$$\sigma \models E(X, Y, Y') \text{ be a counterexample to fix.}$$

- Let $\sigma := \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 1, y_2 \mapsto 1, y'_1 \mapsto 0, y'_2 \mapsto 0\}$.
- Potential repair candidates: All y_i where $\sigma[y_i] \neq \sigma[y'_i]$.

Repair Candidate Identification

$$E(X, Y, Y') := \varphi(X, Y) \wedge \neg\varphi(X, Y') \wedge (Y' \leftrightarrow F(X))$$

$\sigma \models E(X, Y, Y')$ be a counterexample to fix.

- Let $\sigma := \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 1, y_2 \mapsto 1, y_1' \mapsto 0, y_2' \mapsto 0\}$.
- Potential repair candidates: All y_i where $\sigma[y_i] \neq \sigma[y_i']$.
- $\varphi(X, Y)$ is Boolean Relation.
  - So it can be $\hat{\sigma} = \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 0, y_2 \mapsto 1, y_1' \mapsto 0, y_2' \mapsto 0\}$
  - We would not repair f_1.

Repair Candidate Identification

$$E(X, Y, Y') := \varphi(X, Y) \land \neg\varphi(X, Y') \land (Y' \leftrightarrow F(X))$$

$$\sigma \models E(X, Y, Y') \text{ be a counterexample to fix.}$$

- Let $\sigma := \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 1, y_2 \mapsto 1, y'_1 \mapsto 0, y'_2 \mapsto 0\}$.
- Potential repair candidates: All y_i where $\sigma[y_i] \neq \sigma[y'_i]$.
- $\varphi(X, Y)$ is Boolean Relation.
  - So it can be $\hat{\sigma} = \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 0, y_2 \mapsto 1, y'_1 \mapsto 0, y'_2 \mapsto 0\}$
  - We would not repair f_1.
- MaxSAT-based Identification of *nice counterexamples*:
  - Hard Clauses $\varphi(X, Y) \land (X \leftrightarrow \sigma[X])$.
  - Soft Clauses $(Y \leftrightarrow \sigma[Y'])$.
- Candidates to repair: Y variables in the violated soft clauses

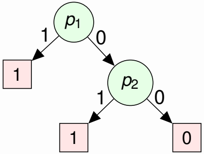
Repairing Approximations

- $\sigma = \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 0, y_2 \mapsto 1, y_1' \mapsto 0, y_2' \mapsto 0\}$, and we want to repair f_2.
- Potential Repair: If $\underbrace{x_1 \wedge x_2 \wedge \neg y_1}_{\beta=\{x_1, x_2, \neg y_1\}}$ then $y_2 = 1$
- Would be nice to have $\beta = \{x_1, x_2\}$ or even $\beta = \{x_1\}$
- Challenge: How do we find small β?
  - $G_\sigma(X, Y) := \varphi(X, Y) \wedge x_1 \wedge x_2 \wedge \neg y_1 \wedge (y_2 = 0)$
  - β:= Literals in UNSAT Core of $G_\sigma(X, Y)$

Repair: Adding Level to Decision List

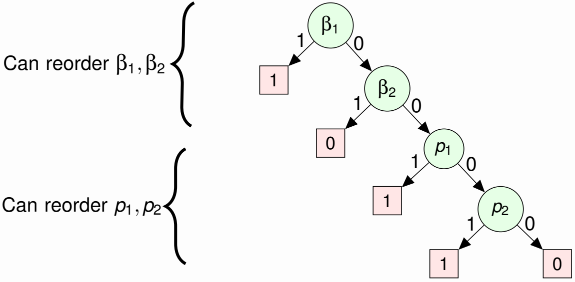
- Candidates are from one level decision list:
  - Say we have paths p_1, p_2 with the leaf node label as 1.
  - Learned decision tree: If p_1 then 1, elif p_2 then 1, else 0.
  - p_1, p_2 can be reordered.

Can reorder p_1, p_2



Repair: Adding Level to Decision List

- Candidates are from one level decision list:
  - Say we have paths p_1, p_2 with the leaf node label as 1.
  - Learned decision tree: If p_1 then 1, elif p_2 then 1, else 0.
  - p_1, p_2 can be reordered.
- Suppose in repair iterations, we have learned: If β_1 then 1, $\ldots \beta_2$ then 0 ......
- β_1 and β_2 can be reordered.
- From one-level decision list to two-level decision list.

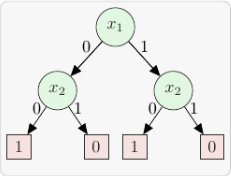


$\varphi(X, Y)$
$X = \{x_1, x_2\}$
$Y = \{y_1, y_2\}$

Data Generation →

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	0
1	1	0	1

Learn Candidates →



Verify Candidates ↓

Check Satisfiability of $E(X, Y, Y')$

SAT, σ → $G_\sigma(X, Y)$

UNSAT Core-based Repair → Check Satisfiability

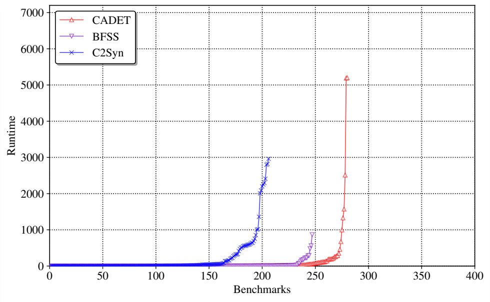
UNSAT ↓

Return F

Experimental Evaluations

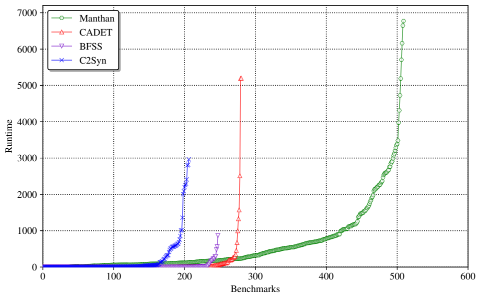
- 609 Benchmarks from:
  - QBFEval competition
  - Arithmetic
  - Disjunctive decomposition
  - Factorization
- Compared Manthan with State-of-the-art tools: CADET (Rabe et al., 2019), BFSS (Akshay et al. ,2018), C2Syn (Chakraborty et al., 2019).
- Timeout: 7200 seconds.

Experimental Evaluations



C2Syn	BFSS	CADET
206	247	280

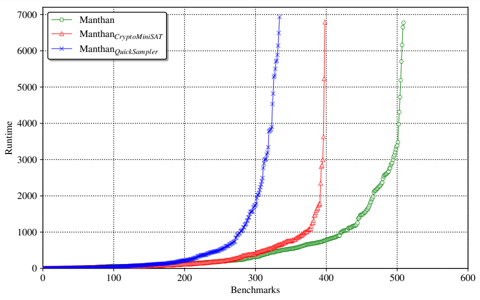
Experimental Evaluations



C2Syn	BFSS	CADET	Manthan
206	247	280	509

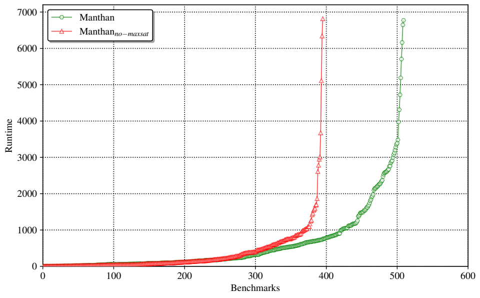
An increase of 229 benchmarks.

Impact of Choices (I): Data Generation



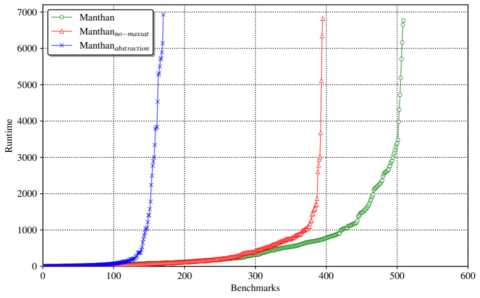
QuickSampler	CryptoMiniSAT	CMSGen
332	399	509

Impact of Choices (II): Use of MaxSAT



Manthan$_{\text{no-maxsat}}$	Manthan
396	509

Impact of Choices (III): Abstraction vs Approximation



Manthan$_{abstraction}$	Manthan$_{no\text{-}maxsat}$	Manthan
171	396	509

Program Synthesis: Experimental Evaluation

609 bit-vector instances from SyGuS competition

Syntax-Guided Solvers						
DryadSynth	Stochpp	Symbolic	ESolver	EUSolver	CVC4	Manthan
15	39	108	151	236	488	592

Future work: Interesting Questions

- From Abstraction to Approximations in Verification?
- Beyond proposition synthesis: SMT
- Learning Theoretic Foundations for Functional Synthesis
  - What is the ideal distribution to generate the data?
  - Mistake bounds/complexity of learning functions from relations?
- The Future of Formal Methods (FM) +Machine Learning (ML)
  - The proposed solutions by ML do not need to be fully correct.
  - Use FM for correctness and ML to quickly find the solution.

Conclusion

Manthan: A Data-Driven Approach for Boolean Functional Synthesis.

- Constrained Sampling
- Decision List Classifier
- Formal Methods

Solves 509 benchmarks — state of the art could solve 280

https://github.com/meelgroup/manthan

Thanks!

Manthan: Example

- Let $X = \{x_1, x_2\}$, and $Y = \{y_1, y_2\}$
- $\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \vee x_2)) \wedge (y_2 \leftrightarrow (x_1 \wedge (x_2 \vee y_1)))$
- Skolem Functions:
  - $f_1(x_1, x_2) := (x_1 \vee x_2)$
  - $f_2(x_1, x_2, y_1) := (x_1 \wedge (x_2 \vee y_1))$
    $f_2(x_1, x_2, y_1) := (x_1 \wedge (x_2 \vee (x_1 \vee x_2))$
    $f_2(x_1, x_2, y_1) := x_1$

$$\exists Y \varphi(X, Y) \equiv \varphi(X, F(X))$$

Example: Data Generation

Let $X = \{x_1, x_2\}$, and $Y = \{y_1, y_2\}$

$$\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \vee x_2)) \wedge (y_2 \leftrightarrow (x_1 \wedge (x_2 \vee y_1)))$$

Constrained Sampler

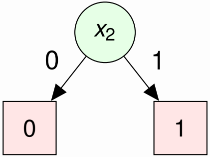
x_1	x_2	y_1	y_2
0	0	0	0
0	1	1	0
1	1	1	1

Example: Learning Candidate Functions

$$\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \vee x_2)) \wedge (y_2 \leftrightarrow (x_1 \wedge (x_2 \vee y_1)))$$

- Learn candidate function f_1.
- Feature set for $y_1 := \{x_1, x_2\}$

x_1	x_2	y_1
0	0	0
0	1	1
1	1	1



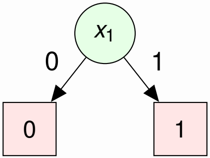
$$f_1(x_1, x_2) := x_2$$

Example: Learning Candidate Functions

$$\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \lor x_2)) \land (y_2 \leftrightarrow (x_1 \land (x_2 \lor y_1)))$$

- Learn candidate function f_2.
- Feature set for y_2 := $\{x_1, x_2, y_1\}$

x_1	x_2	y_1	y_2
0	0	0	0
0	1	1	0
1	1	1	1



$$f_2(x_1, x_2, y_1) := x_1$$

Example: Verification of Candidate Functions

$$\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \vee x_2)) \wedge (y_2 \leftrightarrow (x_1 \wedge (x_2 \vee y_1)))$$

- $E(X, Y, Y') := \varphi(X, Y) \wedge \neg\varphi(X, Y') \wedge (Y' \leftrightarrow F(X))$

$$E(X, Y, Y') := \varphi(x_1, x_2, y_1, y_2) \wedge \neg\varphi(x_1, x_2, y'_1, y'_2) \wedge (y'_1 \leftrightarrow x_2) \wedge (y'_2 \leftrightarrow x_1)$$

$\downarrow$

SAT

$\downarrow$

$\sigma \models E(X, Y, Y') \longrightarrow \sigma[x_1] = 1, \sigma[x_2] = 0$

$\swarrow \quad \searrow$

$\sigma[y_1] = 1,\ \sigma[y_2] = 1 \qquad \sigma[y'_1] = 0,\ \sigma[y'_2] = 1$

$$\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \vee x_2)) \wedge (y_2 \leftrightarrow (x_1 \wedge (x_2 \vee y_1)))$$

- $E(X, Y, Y') := \varphi(X, Y) \wedge \neg\varphi(X, Y') \wedge (Y' \leftrightarrow F(X))$

$$E(X, Y, Y') := \varphi(x_1, x_2, y_1, y_2) \wedge \neg\varphi(x_1, x_2, y'_1, y'_2) \wedge (y'_1 \leftrightarrow x_2) \wedge (y'_2 \leftrightarrow x_1)$$

$\downarrow$

SAT

$\downarrow$

$\sigma \models E(X, Y, Y') \longrightarrow \sigma[x_1] = 1, \sigma[x_2] = 0$

$\sigma[y_1 = 1], \sigma[y_2] = 1$ | $\sigma[y'_1 = 0], \sigma[y'_2] = 1$

$\sigma[y_1] \neq \sigma[y'_1]$

Candidate to repair f_1

Example: Repairing candidate functions (I)

$$\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \vee x_2)) \wedge (y_2 \leftrightarrow (x_1 \wedge (x_2 \vee y_1)))$$

- $G_1(X, Y) = \varphi(X, Y) \wedge (X \leftrightarrow \sigma[X]) \wedge (y_1 \leftrightarrow \sigma[y_1']]$.
- $G_1(X, Y) = \varphi(X, Y) \wedge (x_1 \leftrightarrow 1) \wedge (x_2 \leftrightarrow 0) \wedge (y_1 \leftrightarrow 0)$.
- UNSAT core of $G_1(X, Y) = \varphi(X, Y) \wedge (x_1 \leftrightarrow 1) \wedge (y_1 \leftrightarrow 0)$
- Repair formula $\beta = x_1$.

Example: Repairing candidate functions (II)

$$\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \vee x_2)) \wedge (y_2 \leftrightarrow (x_1 \wedge (x_2 \vee y_1)))$$

Before repair	Repair	After repair
$f_1(\sigma[X]) \mapsto 0$	$f_1(X) \leftarrow f_1(X) \vee \beta$ $f_1(X) \leftarrow x_2 \vee x_1$	$f_1(X) \mapsto 1$

Example: Verification of Candidate Functions

$$\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \vee x_2)) \wedge (y_2 \leftrightarrow (x_1 \wedge (x_2 \vee y_1)))$$

- $E(X, Y, Y') := \varphi(X, Y) \wedge \neg\varphi(X, Y') \wedge (Y' \leftrightarrow F(X))$

$$E(X, Y, Y') := \varphi(x_1, x_2, y_1, y_2) \wedge \neg\varphi(x_1, x_2, y'_1, y'_2) \wedge (y'_1 \leftrightarrow x_2 \vee x_1) \wedge (y'_2 \leftrightarrow x_1)$$

↓

UNSAT

↓

Manthan returns *F*

Data Generation

- $\Sigma_1 :=$ Sample 500 data point with $W(x_i) = 0.5$ and $W(y_i) = 0.9$.

$$w_1(i) = \frac{\text{Count}(\Sigma_1 \cap (y_i = 1))}{500}$$

- $\Sigma_2 :=$ Sample 500 data point with $W(x_i) = 0.5$ and $W(y_i) = 0.1$.

$$w_2(i) = \frac{\text{Count}(\Sigma_2 \cap (y_i = 0))}{500}$$

- If $0.35 < w_1(i) < 0.65$ and $0.35 < w_2(i) < 0.65$, then $q_i = w_1(i)$, else $q_i = 0.9$.